Esitehtävät

1.

a) Tuottaja-kuluttaja-ongelmassa on kyse kahden eri prosessin (tuottajan ja kuluttajan) kommunikaatiosta. Tuottajaprosessin täytyy jollakin tavalla välittää tuottamansa data kuluttajaprosessille. Tämä tapahtuu puskuritaulukon kautta. Koska puskuritaulukko on rajallinen, täytyy tuottajaprosessin odottaa silloin kun taulukko on täynnä. Kuluttajaprosessi odottaa, kun taulukko on tyhjä. Ongelma syntyy prosessien nopeudesta, jos esimerkiksi kuluttaja on huomattavasti tuottajaa hitaampi prosessi, voi puskuritaulukko täyttyä, jolloin tuottaja joutuu odottamaan, että kuluttajaprosessi saa vapautettua tilaa taulukosta. Jos kuluttaja tyhjentää taulukkoa nopeammin kuin tuottaja tuottaa sinne dataa, joutuu kuluttajaprosessi odottamaan, että taulukkoon tulee kulutettavaa dataa.

b) Kriittisen alueen ongelma ilmenee silloin kun monta eri prosessia yrittää muokata jaettua dataa yhtäaikaa. Esimerkiksi prosessi 1 kertoo muuttujan x arvon luvulla 2 ja prosessi 2 kertoo saman muuttujan arvon samalla luvulla. Lopputuloksen täytyisi siis olla 4*x, mutta jos prosessit lukevat arvon x samaan aikaan, molemmat tallentavat lopuksi arvon 2*x.

c) Lukkiutuma syntyy, kun esimerkiksi prosessi 1 odottaa, että toiset prosessit vapauttaisivat resursseja, joita se tarvitsee, mutta kyseiset prosessit odottavat resursseja prosessilta 1. Ongelma johtuu siitä, että tietokoneella ei ole rajattomasti muistia käytössään.

2.

Ratkaisu 1.

Kun (in+1)mod BUFFER_SIZE == out kaikki muistipaikat ovat täynnä eikä tuottajaprosessi tuota mitään. Muulloin tuottajaprosessi lisää puskuritaulukkoon in-muuttujan osoittamalle paikalle uuden tuotetun datan, jonka jälkeen tuottaja muuntaa in-muuttujan arvoon (in + 1) % BUFFER_SIZE eli siirtyy yhden muistipaikan eteenpäin siirtyen muistipaikkaan 0 arvon BUFFER_SIZE-1 (10 – 1 = 9) jälkeen, koska ((9 + 1) % 10 = 0 . Kuluttajaprosessi taas ei tee mitään silloin kun in-muuttuja on sama kuin out-muuttuja eli kun puskuritaulukossa ei ole mitään kulutettavaa (kuluttaja on kuluttanut tuottajan viimeisimmän tuotetun arvon). Muulloin kuluttaja kuluttaa puskuritaulukosta arvon out-muuttujan osoittamalta paikalta, jonka jälkeen se muuntaa out-muuttujan arvoon (out + 1) % BUFFER_SIZE eli siirtyy yhden muistipaikan eteenpäin siirtyen muistipaikkaan 0 samoin kuin in-muuttuja.

Ratkaisu 2.

Tässä ratkaisussa laskuri (counter) ylläpitää tietoa paljonko puskurissa on dataa odottamassa. Tuottaja ei tuota mitään silloin kun counter = BUFFER_SIZE. Muulloin tuottaja tuottaa puskuritaulukkoon in-muuttujan osoittamalle paikalle uuden arvon, jonka jälkeen tuottaja muuntaa in-muuttujan arvoon (in + 1) % BUFFER_SIZE kuten edellisessä ratkaisussa. Tämän jälkeen tuottaja nostaa counter-muuttujan arvoa yhdellä, mikä tarkoittaa että puskuritaulukossa on nyt yksi arvo lisää. Tuottajaprosessi ei tee mitään, kun counter = 0 eli kun puskuritaulukko on tyhjä. Muulloin kuluttaja kuluttaa arvon puskuritaulukosta out-muuttujan osoittamalta paikalta, jonka jälkeen se muuntaa out-muuttujan edellisen ratkaisun tavoin arvoon (out + 1) % BUFFER_SIZE. Tämän jälkeen kuluttaja vielä vähentää counter-muuttujan arvoa yhdellä, mikä tarkoittaa, että puskuritaulukosta on käyty yksi arvo läpi.

3.

a) Keskinäinen poissulkeminen, ilmoitus lipulla, Petersonin algoritmi, leipomoalgoritmi

b) Test-and-set, Compare-and-Swap

c) Semaforit, Monitorit, Solaris synkronointi, Windows synkronointi, Linux synkronointi, Pthreads synkronointi

4.

Varmistetaan, että lukkiutumaa ei tapahdu.

Esimerkiksi tunnistetaan lukkiutumavaaran syntyminen, eikä allokoida niitä resursseja, jotka johtaisivat lukkiutumaan. Kun prosessi pyytää resursseja, arvioidaan syntyvää tilannetta, jos resurssit allokoitaisiin prosessille ja arvion lopputuloksesta riippuen tehdään päätös resurssien allokoinnista.

Sallitaan lukkiutuma, tunnistetaan kun se tapahtuu ja toivutaan siitä jollain menetelmällä.

Esimerkiksi resurssit voidaan viedä prosesseilta ja jakaa uudelleen, jotta lukkiutuma purkautuu.

Ei välitetä lukkiutumasta. Esitetään, että ongelmaa ei ole.

5.

a) Fork luo identtisen kopion äitiprosessista (lapsiprosessin) ja palauttaa äitiprosessille lapsiprosessin PID-numeron.

b) Koska lapsiprosessi on äitiprosessin identtinen kopio, se perii äitiprosessilta kaiken paitsi PID-numeronsa.

c) Fork-komento voi epäonnistua, jos järjestelmän yhtäaikaisten prosessien maksimimäärä on saavutettu tai jos yksittäisen käyttäjän yhtäaikaisten prosessien maksimäärä on saavutettu.

6.

a) Exec lataa lapsiprosessin muistiavaruuteen uuden ohjelman.

b) Exec-komento korvaa prosessin uudella prosessilla, jolloin alkuperäisen prosessin säikeet katkeavat eli ne tuhoutuvat.

7.

a) Socket-komento luo pistokkeen.

b) Bind-komento sitoo pistokkeen paikalliseen pistokeosoitteeseen.